USING THE IDT79R3051™ AND THE IDT79R3081™ WITH THE HP16500 LOGIC ANALYZER

APPLICATION NOTE AN-111

Integrated Device Technology, Inc.

Supplement to Application Note AN-93

By Gary Szilagyi

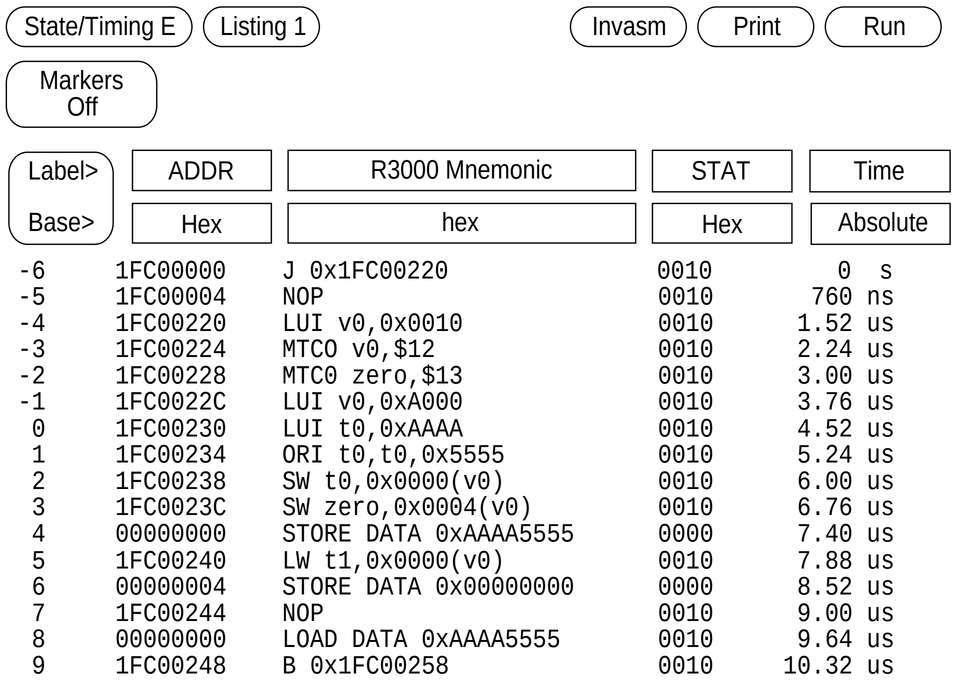
INTRODUCTION

In Application Note-93, the use of IDT's 7RS364 disassembler with the HP16500 Logic Analyzer for the IDT79R3051™ RISController™ family of CPUs was discussed in detail. However, the original versions of the disassembler were form-fitted for the R3000 CPU interface of a 32-bit non-multiplexed bus design. In order to accommodate the high level of integration on-board the R3051, including the 4kB–8kB of instruction cache, 2kB of data cache, 4-deep read and write buffers and the R3000A execution engine—all in a single 84-pin package, the 32-bit bus required multiplexing address and data pins. Although the original versions of the disassembler remain compatible with the new family of IDT's RISControllers, an effort was made to simplify the interface between R3051 and the disassembler to accommodate simple triggering schemes, as well as future IDT embedded controllers that continue in the path of the R3051 family.

THE IDT7RS364 DISASSEMBLER AND THE IDTR3051

The IDT7RS364 Disassembler consists of a software package that greatly eases the task of debugging software on the IDTR3051 family of CPUs. The HP16500 allows the capture of executed hex/binary machine opcodes in a typical Logic Analyzer State Trace Listing format with the ability to decode and display the acquisitions in the R3000 assembly code mnemonic format, as seen in Figure 1. Thus, the engineer does not have to resort to look-up tables, and can effectively determine the exact processor state for easy software debugging.

The original versions of the disassembler were form-fitted to the R3000 CPU interface. Although the derivative products of the IDT R3051 family are compatible, the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals used for data acquisitions by the disassembler package causes some confusion during a high-speed burst read. As discussed in Application Note AN-93, the work-around was to create a more complex read strobe in order to capture a four- word burst read by setting up a trigger mechanism on the HP16500 that looks like: [(SysClk ==↑) AND [($\overline{\text{ACK}}$ == 0) OR $\overline{\text{RDCEN}}$ == 0)]. However, this is only applicable to systems that bring the $\overline{\text{ACK}}$ signal LOW at precisely the same time the $\overline{\text{RDCEN}}$ is LOW, or that don't bring it LOW at all during a four word burst read. If, for instance, the $\overline{\text{ACK}}$ signal triggered in the phase between two successive $\overline{\text{RDCEN}}$s, a duplicated capture would occur. The disassembler was modified a second time to remedy this situation. In a read cycle, the $\overline{\text{RD}}$ pin will be asserted LOW for the entire cycle and the $\overline{\text{RDCEN}}$ signal toggles to successfully pass each of the four words across the bus. The newest version of the disassembler



```
-6     1FC00000    J 0x1FC00220                0010        0  s
-5     1FC00004    NOP                         0010      760 ns
-4     1FC00220    LUI v0,0x0010               0010     1.52 us
-3     1FC00224    MTCO v0,$12                 0010     2.24 us
-2     1FC00228    MTC0 zero,$13               0010     3.00 us
-1     1FC0022C    LUI v0,0xA000               0010     3.76 us
 0     1FC00230    LUI t0,0xAAAA               0010     4.52 us
 1     1FC00234    ORI t0,t0,0x5555            0010     5.24 us
 2     1FC00238    SW t0,0x0000(v0)            0010     6.00 us
 3     1FC0023C    SW zero,0x0004(v0)          0010     6.76 us
 4     00000000    STORE DATA 0xAAAA5555       0000     7.40 us
 5     1FC00240    LW t1,0x0000(v0)            0010     7.88 us
 6     00000004    STORE DATA 0x00000000       0000     8.52 us
 7     1FC00244    NOP                         0010     9.00 us
 8     00000000    LOAD DATA 0xAAAA5555        0010     9.64 us
 9     1FC00248    B 0x1FC00258                0010    10.32 us
```

Figure 1. R3051 Address/Data Trace List on a Logic Analyzer

begins "LOAD" captures not on $\overline{\text{RD}}$, but rather upon the $\overline{\text{RDCEN}}$. For interleaved memory systems that do not toggle the $\overline{\text{RDCEN}}$ pin, please refer to section "Hazards" for more details. During a write cycle, it triggers upon the rising edge (from LOW-to-HIGH) of the $\overline{\text{WR}}$ signal. Thus, the newest revision of the disassembler now expects the $\overline{\text{RDCEN}}$ and the $\overline{\text{WR}}$ signals as clocks to strobe the address and data into the HP16500, as well as the $\overline{\text{WR}}$, DIAG_1 and DIAG_0 to verify and decode the processor status

INTERFACING THE HP16500 TO THE '385 EVALUATION BOARD

In order to insure proper operation of the disassembler, the correct interface between the R305x target system and the HP16500 must be available. The disassembler requires a particular pinout setup on the logic analyzer's five 16-channel probe pod sets. The interface protocol must be followed for correct interpretation of the address, data, and status lines by the pre-processor. Table 1 displays the default pod connections that the HP16500 expects (same setup for the 7RS385 evaluation board). This information is stored on disk in the configuration file "DIS_305x_E". When loaded, this file not only loads the disassembler, but also all the state and timing information, including the default pod connections expected at the system interface.

Application Note-93 discusses in detail the interface between typical R305x based systems and the logic analyzer. Rather than repeat that discussion, the interface between the 7RS385 Evaluation board and the disassembler requires some elaboration. For instance, the '385 Hardware User's Manual shows the connections to be made from the board's five 20-pin logic analyzer sockets and the logic analyzer's five, 16-channel pods. Note however that in section 2–5 of the '385 Hardware User's Manual, the connections on the status pod (pod#5) are incorrect. In order to be consistent with the protocol of the disassembler, some of the pins need to be connected as follows:

- WR (J12 pin #17) needs to be on pod #5 channel #4
- RDCEN (J12 pin #14) needs to be on pod #5 channel #5

The disassembler also requires status lines for determining processor status: $\overline{\text{WR}}$, $\overline{\text{RDCEN}}$, DIAG_1, and DIAG_0. The $\overline{\text{WR}}$ signal distinguishes between read and write cycles. The $\overline{\text{RDCEN}}$ pin is used to identify a false trigger for applications that assert the $\overline{\text{RDCEN}}$ signal during writes. In order to avoid a duplicate capture, the $\overline{\text{RDCEN}}$ signal is polled to determine if it was the cause of the acquisition. If it was, then a trigger-

Table 1. R3051 Default Pod Connections on the HP16500 Logic Analyzer

POD	5	POD	4	POD	3	POD	2	POD	1
chan	sig	chan	sig	chan	sig	chan	sig	chan	sig
15	X	15	A/D(31)	15	A/D(15)	15	A(31)	15	A(15)
14	X	14	A/D(30)	14	A/D(14)	14	A(30)	14	A(14)
13	X	13	A/D(29)	13	A/D(13)	13	A(29)	13	A(13)
12	Diag_1[(2)]	12	A/D(28)	12	A/D(12)	12	A(28)	12	A(12)
11	X	11	A/D(27)	11	A/D(11)	11	A(27)	11	A(11)
10	Diag_0	10	A/D(26)	10	A/D(10)	10	A(26)	10	A(10)
9	X	9	A/D(25)	9	A/D(9)	9	A(25)	9	A(9)
8	X	8	A/D(24)	8	A/D(8)	8	A(24)	8	A(8)
7	X	7	A/D(23)	7	A/D(7)	7	A(23)	7	A(7)
6	X	6	A/D(22)	6	A/D(6)	6	A(22)	6	A(6)
5	$\overline{\text{RDCEN}}$	5	A/D(21)	5	A/D(5)	5	A(21)	5	A(5)
4	$\overline{\text{WR}}$	4	A/D(20)	4	A/D(4)	4	A(20)	4	A(4)
3	X	3	A/D(19)	3	A/D(3)	3	A(19)	3	Addr(3)
2	X	2	A/D(18)	2	A/D(2)	2	A(18)	2	Addr(2)
1	X	1	A/D(17)	1	A/D(1)	1	A(17)	1	$\overline{\text{BEN(1)}}$
0	X	0	A/D(16)	0	A/D(0)	0	A(16)	0	$\overline{\text{BEN(2)}}$
NClk	$\overline{\text{WR}}$	MClk	$\overline{\text{RDCEN}}$	LClk		KClk		JClk	

NOTES:

1. Master Clock Format: N↑ + M↑ (default for the 7RS385 Evaluation Board setup)
2. POD5(12) is Diag_1 and POD5(10) is Diag_0 (Diag pins are not latched on the 7RS385 Eval Board). If running uncached, then Diag_1 MUST be grounded (GND), and Diag_0 is not used by disassembler.
3. A(31:4) are connected to the Address Latch outputs. The rest of the signals are connected to R3051 outputs. X's denote unused probes that can be assigned by the user.

error message, “T.E”, and the store instruction along with the write data on the bus is displayed (e.g. "T.E. (STORE 0xxxxxxxxx)). The diagnostic pin DIAG_1 distinguishes if the external memory read was cacheable, and if so, determines with DIAG_0 if it was an instruction or data read. Note that for the newest IDT embedded controller, the R3081, DIAG_1 is defined during writes, yielding cache information for “STORE” instructions. A second version of the disassembler, “DIS_3081”, exploits this feature for external cache support. By defining the DIAG_1 pin during writes, the CPU will signal whether the data being written was retained in the on-chip data cache. Keep in mind that the DIAG_0 pin remains undefined during write cycles. This information is extremely helpful to the programmer to determine the processor's state when tracing through the software.

The diagnostic pins on the '385 board are **NOT LATCHED**, and therefore are time-multiplexed pins. Thus, the user must either latch these pins with an external latch as seen in Figure 2 or proper decoding of cached code, or connect both diagnostic pins to GND. Although the disassembler is capable of interpreting the bus transactions of cached code, keep in mind that all logic analyzers and disassemblers can only capture external CPU memory accesses. The R3051 has large internal caches, and is capable of running much of its code from within. In order for the disassembler to accurately reflect the entire instruction/data flow, the R3051 must be ran uncached. For more information regarding running cached code and data, please refer to Application Note AN-93 for a complete discussion.

LOADING AND RUNNING THE DISASSEMBLER

Included in the software package are two files. The first is the disassembler application "DIS_305x". The second is the setup file, "DIS_305x_E", containing all the state and timing information required by the disassembler, as well as the assigned pod connections expected by the HP16500 for the R305x target system.

After the HP operating system boots up completely, the system configuration screen as shown in Figure 3 should be displayed. To load the disassembler into the HP16500, the following steps must be taken:

1. Insert the disassembler diskette into the front disk drive.
2. Select the “Configuration” field as shown in Figure 3**.** A pop-up menu with options will appear. Choose the “Front Disk” under the pop-up menu.
3. A new screen will appear that looks like Figure 4. Select the “Load” and “State/Timing” fields, and load in the configuration file “Dis_305x_E” by selecting “Execute” as shown in Figure 4.

The HP16500 will then load the disassembler, as well as all the state and timing information and the expected pin-configuration as shown in Table 1 previously. Once the disassembler application and setup files are loaded into the HP, the logic analyzer is ready to set trace conditions for data acquisition.

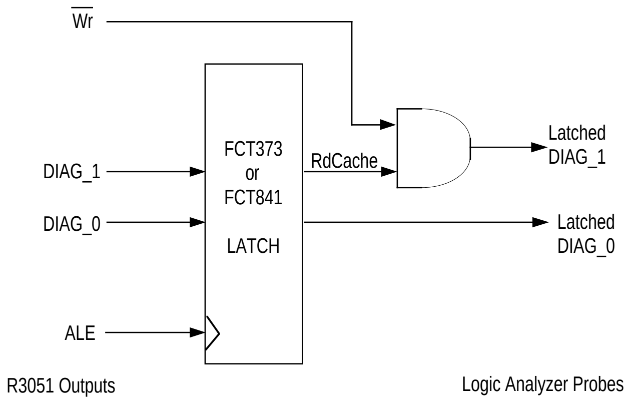


Figure 2. R3051 Address/Data Trace List on a Logic Analyzer

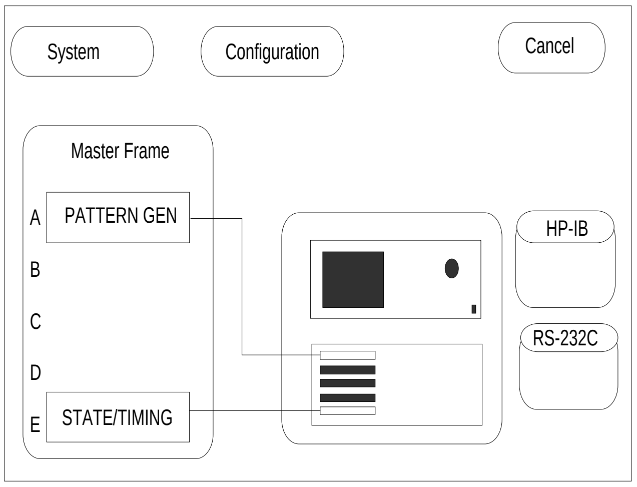
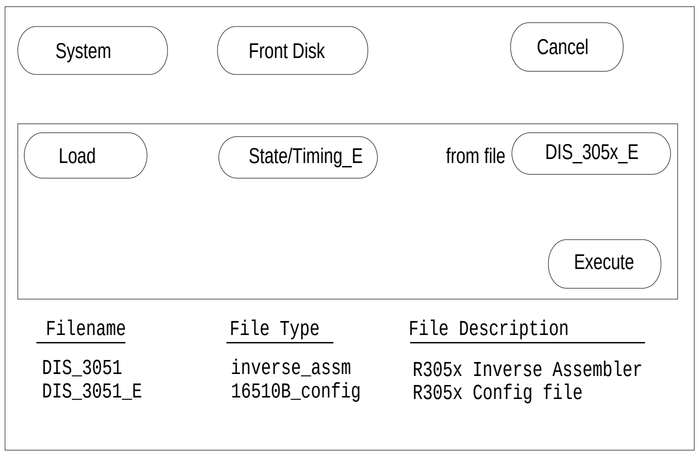


Figure 3. HP16500 Screen Display



Filename	File Type	File Description
DIS_3051	inverse_assm	R305x Inverse Assembler
DIS_3051_E	16510B_config	R305x Config file

Figure 4. HP16500 Load Screen Display

With the application files loaded, the disassembler is almost ready to be triggered by the target system. Follow the steps below that describe how to run and trigger the disassembler package:

1. Select the "System" field as shown in Figure 4. A pop-up menu will appear with the option of "State/Timing". Choose this field to enter the state and timing mode of acquisition.
2. A new window will appear that is shown in Figure 5. Under the "Configuration" menu lies options that allow the user to set display or change the current configuration of the interface, clocks, and pod connections.
3. Trigger the HP16500.

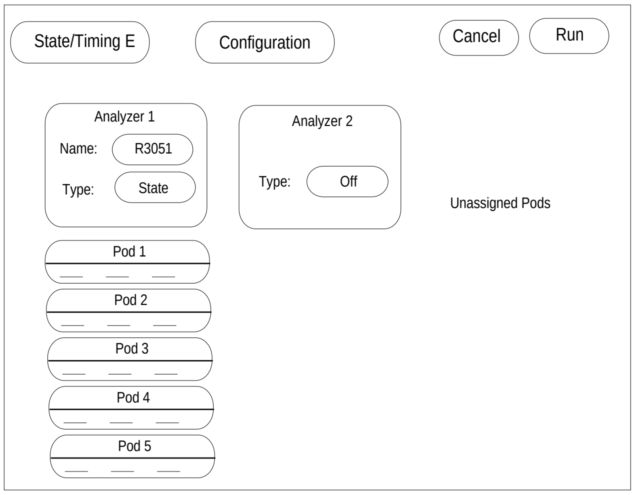
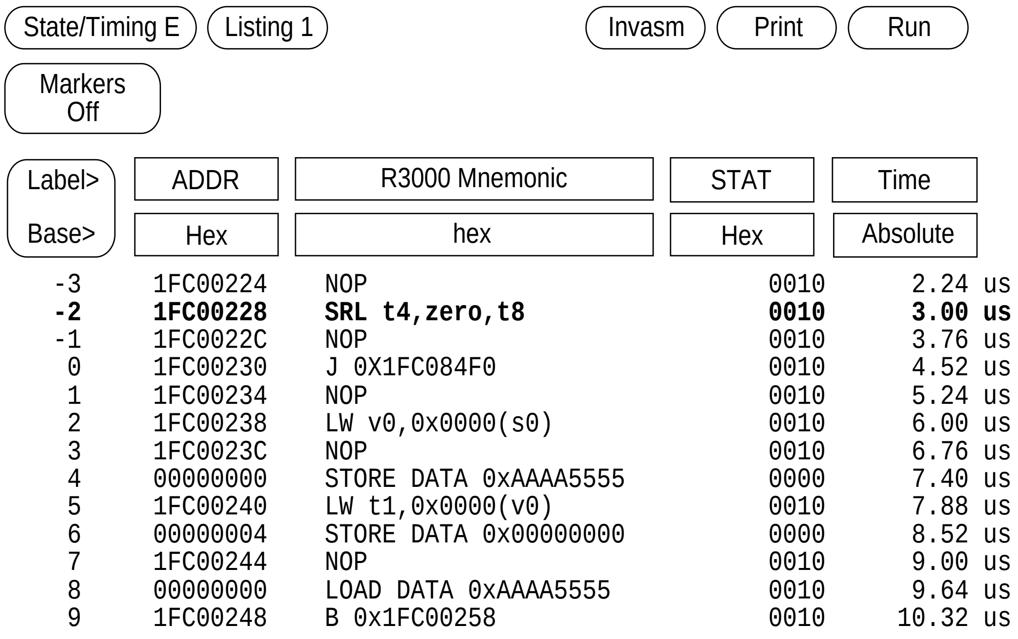


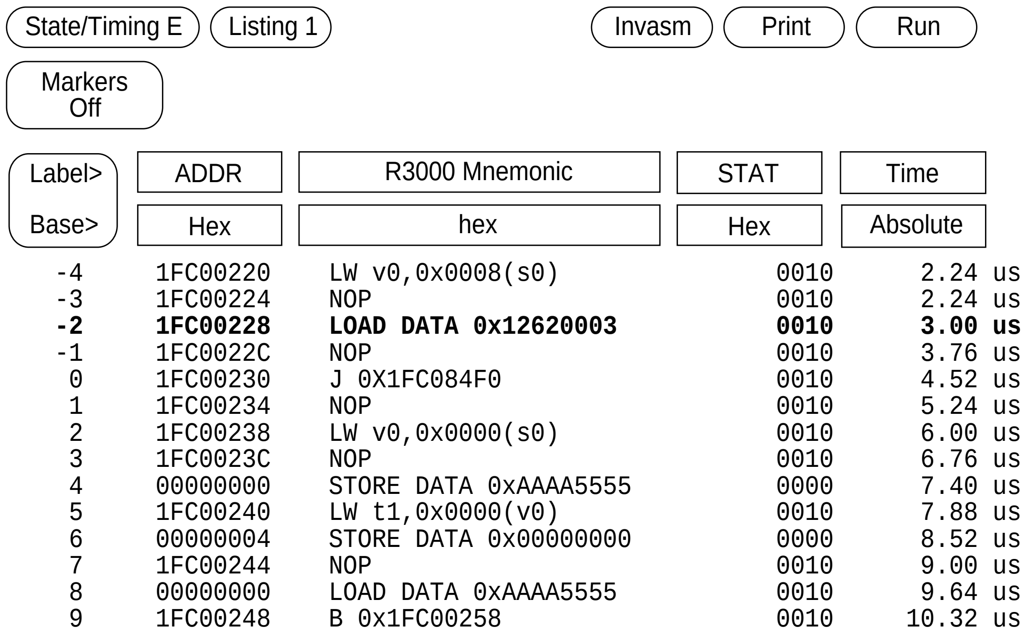
Figure 5. HP16500 State/Timing Mode Display

Once triggered, the logic analyzer will begin its acquisition, and go directly to the "Listing" field. The addresses and disassembled data will be displayed. Note however that the displayed disassembly may be incorrect. This is due to an "unsynchronized" system. The captured data needs to be synchronized with the logic analyzer's display to insure correct disassembly of the bus. The problem of unsynchronized captures arises due to the incomplete status of the processor state for data loads. As a result, when an instruction fetch is scrolled to the top of the screen, and a load data is displayed, but the corresponding load instruction was "cut off" or scrolled off the screen, the disassembler software looses it reference point by which it identifies the load data. As a result, the load data may be decoded incorrectly as an instruction as seen in Figure 6. Notice in this Figure the instruction on line -2. It was disassembled as an instruction instead of as a data load. Also notice the address of the instruction in the sequence of the four word fetch to main memory. This is an unsynchronized display because the corresponding load instruction was scrolled off the top of the display, and due to the way the disassembler interprets and tags the load datas, the reference point was lost. As a result, the load data was interpreted and decoded as an instruction. As shown in Figure 7, the correctly synchronized system has the load instruction displayed at the top of the screen (identified by its address), and the load data is interpreted correctly.



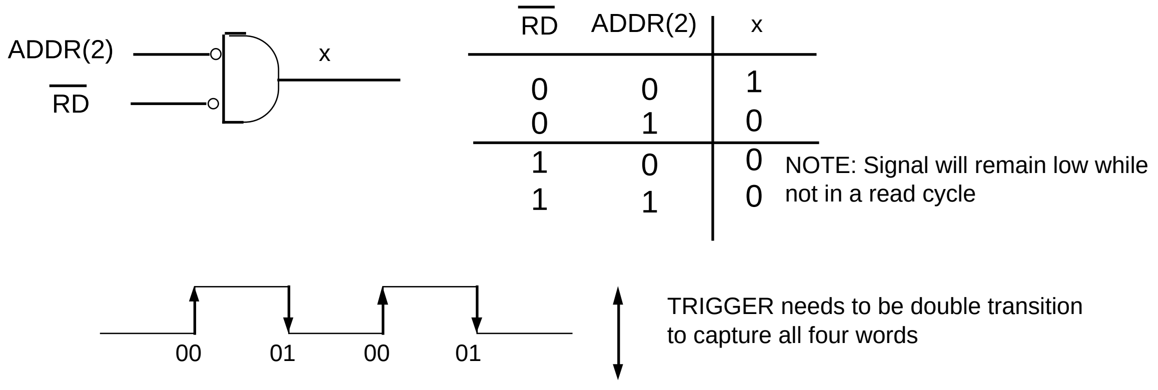
Label>	ADDR	R3000 Mnemonic	STAT	Time
Base>	Hex	hex	Hex	Absolute
-3	1FC00224	NOP	0010	2.24 us
-2	**1FC00228**	**SRL t4,zero,t8**	**0010**	**3.00 us**
-1	1FC0022C	NOP	0010	3.76 us
0	1FC00230	J 0X1FC084F0	0010	4.52 us
1	1FC00234	NOP	0010	5.24 us
2	1FC00238	LW v0,0x0000(s0)	0010	6.00 us
3	1FC0023C	NOP	0010	6.76 us
4	00000000	STORE DATA 0xAAAA5555	0000	7.40 us
5	1FC00240	LW t1,0x0000(v0)	0010	7.88 us
6	00000004	STORE DATA 0x00000000	0000	8.52 us
7	1FC00244	NOP	0010	9.00 us
8	00000000	LOAD DATA 0xAAAA5555	0010	9.64 us
9	1FC00248	B 0x1FC00258	0010	10.32 us

Figure 6. Incorrectly Synchronized Capture (Note line -2)



```
        ADDR        R3000 Mnemonic              STAT        Time
        Hex         hex                         Hex         Absolute
 -4     1FC00220    LW v0,0x0008(s0)            0010         2.24 us
 -3     1FC00224    NOP                         0010         2.24 us
 -2     1FC00228    LOAD DATA 0x12620003        0010         3.00 us
 -1     1FC0022C    NOP                         0010         3.76 us
  0     1FC00230    J 0X1FC084F0                0010         4.52 us
  1     1FC00234    NOP                         0010         5.24 us
  2     1FC00238    LW v0,0x0000(s0)            0010         6.00 us
  3     1FC0023C    NOP                         0010         6.76 us
  4     00000000    STORE DATA 0xAAAA5555       0000         7.40 us
  5     1FC00240    LW t1,0x0000(v0)            0010         7.88 us
  6     00000004    STORE DATA 0x00000000       0000         8.52 us
  7     1FC00244    NOP                         0010         9.00 us
  8     00000000    LOAD DATA 0xAAAA5555        0010         9.64 us
  9     1FC00248    B 0x1FC00258                0010        10.32 us
```

Figure 7. Correctly Synchronized Capture (Note line -2)


ADDR(2)
$\overline{RD}$
x
00
01
00
01
TRIGGER needs to be double transition to capture all four words

$\overline{RD}$	ADDR(2)	x
0	0	1
0	1	0
1	0	0
1	1	0

NOTE: Signal will remain low while not in a read cycle

Figure 8. Simulated $\overline{RDCEN}$ signal

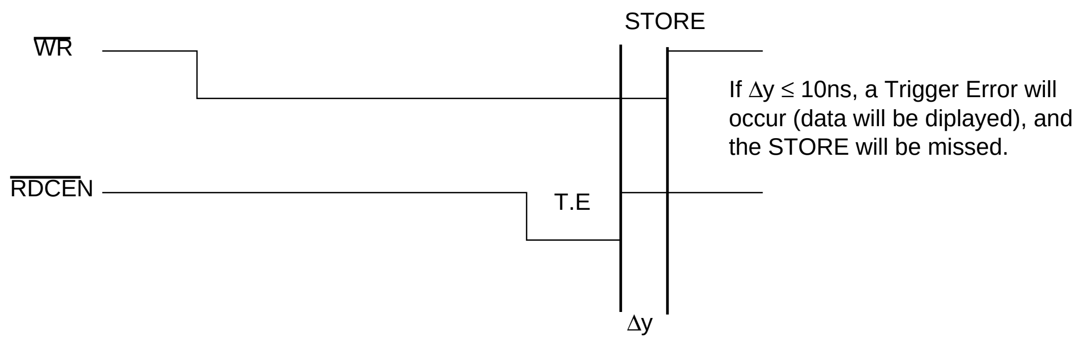

$\overline{WR}$
STORE
If Δy ≤ 10ns, a Trigger Error will occur (data will be diplayed), and the STORE will be missed.
$\overline{RDCEN}$
T.E
Δy

Figure 9. $\overline{RDCEN}$ Asserted during STORE

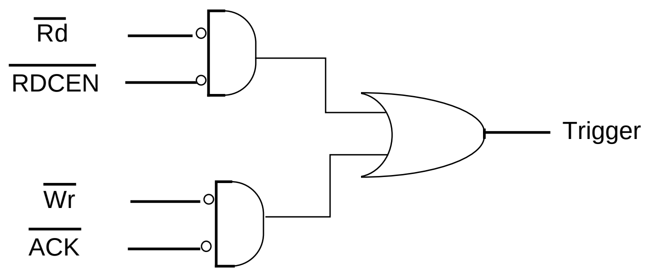


Figure 10. Simple Trigger Logic

To synchronize the system and to insure valid results, the following steps must be taken:

1. Identify the first instruction fetch by its address, not its displayed mnemonic, of the captured data and scroll this line to the top of the screen display.
2. At the top of the HP16500 screen is the field "Ivasm". Select this, and the currently displayed capture will be synchronized.
3. Always make sure that each new capture, or a jump ahead in the analyzer's buffer memory is re-synchronized properly or erroneous data might be displayed. The same applies for any move backwards for any displayed capture.

HAZARDS

For interleaved memory systems that do not toggle the $\overline{\text{RDCEN}}$ four times, but rather keep it asserted, the only data to be captured during quad-word reads will be the last word of the transfer. In order to fix this, the user might wish to simulate a $\overline{\text{RDCEN}}$ strobe during the quad-word read by utilizing the lower order address pins Addr(3:2). This can be accomplished by gating the Addr(2) pin of this 2-bit bus with the $\overline{\text{RD}}$ signal from the CPU. Whenever the next word in the sequence comes across the bus during a read cycle, the transition from LOW-to-HIGH, or HIGH-to-LOW will begin an acquisition, and thus simulate the strobbing of $\overline{\text{RDCEN}}$. Note however, the trigger transition on the HP must be set to both rising and falling transitions as seen in Figure 8**.**

Another hazard to be cautious about is if the $\overline{\text{RDCEN}}$ comes at precisely, or within a 10ns window (Δy) of the rising edge of the $\overline{\text{WR}}$ signal. If so, then this would be regarded as an invalid write with a trigger error (T.E) ocurring and the data on the bus at the time of the invalid capture will be displayed. In this case, the capture on the rising edge of write will be missed and the data displayed with the T.E. is the valid capture as shown in Figure 9. During any case that a $\overline{\text{RDCEN}}$ comes in on a write cycle, a T.E. will occur.

Finally, a feature in HW that would be extremely useful for triggering is a specified trigger signal for the HP logic analyzer that would distinguish between the status of reads and writes triggered by $\overline{\text{ACK}}$. The trigger would simply be established by gating the read and write signals and ORing the results as shown in Figure 10. This should eliminate any trigger edge problems associated with simple data acquisitions for inverse assembly.

SUMMARY

The use of the HP16500 and the IDT7RS364 Disassembler helps to ease the task of software development and debugging on the R305x and the R3081. The disassembler formats logic analyzer state traces into assembly level mnemonics to allow easier user interpretation. It is one of the many useful development tools already available for IDT's MIPS R3000 compatatible CPUs. Similarly, other R3000 software, compilers, as well as other development tools such as the IDT7RS901 IDT/sim ROMable Kernel/Boot Monitor can also be used on R3051 and R3081 systems with little or no modification.